一、数据库基础知识

1.关系型和非关系型数据库的区别?

- 关系型数据库的优点　　MySQL
  - 容易理解。因为它采用了关系模型来组织数据。
  - 可以保持数据的一致性。
  - 数据更新的开销比较小。
  - 支持复杂查询（带where子句的查询）
- 非关系型数据库的优点（NOSQL）　　Redis
  - 不需要经过SQL层的解析，读写效率高。
  - 基于键值对，数据的扩展性很好。
  - 可以支持多种类型数据的存储，如图片，文档等等。

2.SQL语句的分类

DQL：

数据查询语言（凡是带有**select关键字**的都是查询语句）

select...

DML：

数据操作语言（凡是对表当中的数据进行增删改的都是DML）

insert delete update

insert 增

delete 删

update 改

这个主要是操作**表中的数据data。**

DDL：

数据定义语言

凡是带有create、drop、alter的都是DDL。

DDL主要操作的是**表的结构。不是表中的数据。**

create：新建，等同于增

drop：删除

alter：修改

这个增删改和DML不同，这个主要是对表结构进行操作。

TCL：

不是王牌电视。

是事务控制语言

包括：

事务提交：commit;

事务回滚：rollback;

DCL：

是数据控制语言。

例如：授权grant、撤销权限revoke....

二、MySQL相关

MySQL语句执行顺序

```
select
    ...
from
    ...
where
    ...
group by
    ...
order by
    ...
```

以上关键字的顺序不能颠倒，需要记忆。

执行顺序是什么

1.from
2. where
3. group by
4. select
5. order by

MySQL端口

mysql数据库启动的时候，这个服务占有的默认端口号是3306

MySQL中的表

数据库当中最基本的单元是表：table

什么是表table？为什么用表来存储数据呢？

```
姓名        性别        年龄(列：字段)
---------------------------
张三        男            20              -------->行（记录）
李四        女            21              -------->行（记录）
王五        男            22              -------->行（记录）
```

数据库当中是以表格的形式表示数据的。
因为表比较直观。

任何一张表都有行和列：
行（row）：被称为数据/记录。
列（column）：被称为字段。

姓名字段、性别字段、年龄字段。

了解一下：
每一个字段都有：字段名、数据类型、约束等属性。
字段名可以理解，是一个普通的名字，见名知意就行。
数据类型：字符串，数字，日期等。

约束：约束也有很多，其中一个叫做唯一性约束，
这种约束添加之后，该字段中的数据不能重复。

1、查看、创建数据库/表

1.1 进入mysql

```
C:\Users\Administrator>mysql -uroot -p123456
C:\Users\Administrator>mysql -uroot -p
Enter password: ******
```

1.2查看数据库

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
```

1.3使用某个数据库

代码：

```
mysql> use test;
Database changed
```

1.4创建数据库

代码:

```
mysql> create database bjpowernode;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| bjpowernode        |
| mysql              |
| performance_schema |
| test               |
+--------------------+
```

1.5查看当前数据库下有哪些表

代码:

```
mysql> show tables;
```

1.6 查看mysql版本号

代码:

```
mysql> select version();
```

1.7查看当前使用的是哪个数据库

```
mysql> select database();
+-------------+
| database()  |
+-------------+
| bjpowernode |
+-------------+
```

1.8导入数据

```
mysql> source D:\course\03-
MySQL\document\bjpowernode.sql
```

2、查询语句

★2.0 查询语句的执行顺序

语句的执行顺序必须掌握：
第一步：from
第二步：where
第三步：select
第四步：order by（排序总是在最后执行！）

2.1查看表的结构

desc 表名;

2.2查询一个字段

select 字段名 from 表名;

2.3查询多个字段

select deptno,dname from dept;

2.4查询全部数据

select * from 表名

这种方式的缺点：

1、效率低

2、可读性差。

2.5 给表起别名

```
mysql> select deptno,dname as deptname from dept;
+--------+------------+
| deptno | deptname   |
+--------+------------+
|     10 | ACCOUNTING |
|     20 | RESEARCH   |
|     30 | SALES      |
|     40 | OPERATIONS |
+--------+------------+
```

使用as关键字起别名。

注意：只是将显示的查询结果列名显示为deptname，原表列名还是叫：dname

记住：select语句是永远都不会进行修改操作的。（因为只负责查询）

as关键字可以省略，别名不能有空格（如果要有空格要用单引号括起来）

注意：在所有的数据库当中，字符串统一使用单引号括起来，

单引号是标准，双引号在oracle数据库中用不了。但是在mysql

中可以使用。

再次强调：数据库中的字符串都是采用单引号括起来。这是标准的。

双引号不标准。

2.6 select后面可以直接跟*号等数学表达式

```
mysql> select ename,sal*12 from emp; // 结论：字段可以使用数学表达式!
```

2.7 条件查询where

不是将表中所有数据都查出来。是查询出来符合条件的。

语法格式：

```
select
字段1,字段2,字段3....
from
表名
where
条件;
```

条件：

= 等于

<>或!= 不等于

> 大于

< 小于

<= 小于等于

>= 大于等于

between ... and ... 两个值之间, 等同于 >= and <=

is null 为 null（is not null 不为空）

and 并且

or 或者

and和or同时出现，and优先级较高。如果想让or先执行，需要加小括号()

in 包含，相当于多个 or （not in 不在这个范围中） ！！不是区间

not 可以取非，主要用在 is 或 in 中 ,如：is not null 、not in

like 称为模糊查询，支持%或下划线匹配
%匹配任意多个字符
下划线：任意一个字符。
(%是一个特殊的符号，_ 也是一个特殊符号，如果要查询名字中带下划线的字段的话需要加转义字符\)

```
= 等于
查询薪资等于800的员工姓名和编号？
    select empno,ename from emp where sal = 800;
查询SMITH的编号和薪资？
    select empno,sal from emp where ename = 'SMITH';
//字符串使用单引号



<>或!= 不等于
查询薪资不等于800的员工姓名和编号？
    select empno,ename from emp where sal != 800;
    select empno,ename from emp where sal <> 800; //
小于号和大于号组成的不等号



< 小于
查询薪资小于2000的员工姓名和编号？
```

```
    mysql> select empno,ename,sal from emp where sal
< 2000;
    +-------+--------+---------+
    | empno | ename  | sal     |
    +-------+--------+---------+
    |  7369 | SMITH  |  800.00 |
    |  7499 | ALLEN  | 1600.00 |
    |  7521 | WARD   | 1250.00 |
    |  7654 | MARTIN | 1250.00 |
    |  7844 | TURNER | 1500.00 |
    |  7876 | ADAMS  | 1100.00 |
    |  7900 | JAMES  |  950.00 |
    |  7934 | MILLER | 1300.00 |
    +-------+--------+---------+
```

<= 小于等于

查询薪资小于等于3000的员工姓名和编号？

```
    select empno,ename,sal from emp where sal <=
3000;
```

>大于

查询薪资大于3000的员工姓名和编号？

```
    select empno,ename,sal from emp where sal >
3000;
```

>= 大于等于

查询薪资大于等于3000的员工姓名和编号？

```
    select empno,ename,sal from emp where sal >=
3000;
```

between … and …. 两个值之间，等同于 >= and <=

查询薪资在2450和3000之间的员工信息？包括2450和3000

第一种方式：>= and <= （and是并且的意思。）

```
select empno,ename,sal from emp where sal >= 2450 and sal <= 3000;
```

```
+-------+-------+---------+
| empno | ename | sal     |
+-------+-------+---------+
|  7566 | JONES | 2975.00 |
|  7698 | BLAKE | 2850.00 |
|  7782 | CLARK | 2450.00 |
|  7788 | SCOTT | 3000.00 |
|  7902 | FORD  | 3000.00 |
+-------+-------+---------+
```

第二种方式：between … and …

```
select
    empno,ename,sal
from
    emp
where
    sal between 2450 and 3000;
```

注意：

使用between and的时候，必须遵循左小右大。

between and是闭区间，包括两端的值。

is null 为 null（is not null 不为空）

查询哪些员工的津贴/补助为null？

```
mysql> select empno,ename,sal,comm from emp where comm = null;
Empty set (0.00 sec)

mysql> select empno,ename,sal,comm from emp where comm is null;
```

```
    +-------+--------+---------+------+
    | empno | ename  | sal     | comm |
    +-------+--------+---------+------+ |  7369 |
SMITH  |  800.00 | NULL |
    |  7566 | JONES  | 2975.00 | NULL |
    |  7698 | BLAKE  | 2850.00 | NULL |
    |  7782 | CLARK  | 2450.00 | NULL |
    |  7788 | SCOTT  | 3000.00 | NULL |
    |  7839 | KING   | 5000.00 | NULL |
    |  7876 | ADAMS  | 1100.00 | NULL |
    |  7900 | JAMES  |  950.00 | NULL |
    |  7902 | FORD   | 3000.00 | NULL |
    |  7934 | MILLER | 1300.00 | NULL |
    +-------+--------+---------+------+
    10 rows in set (0.00 sec)
```

注意：在数据库当中null不能使用等号进行衡量。需要使用is null

因为数据库中的null代表什么也没有，它不是一个值，所以不能使用

等号衡量。

查询哪些员工的津贴/补助不为null？

```
    select empno,ename,sal,comm from emp where comm
is not null;
    +-------+--------+---------+---------+
    | empno | ename  | sal     | comm    |
    +-------+--------+---------+---------+
    |  7499 | ALLEN  | 1600.00 |  300.00 |
    |  7521 | WARD   | 1250.00 |  500.00 |
    |  7654 | MARTIN | 1250.00 | 1400.00 |
    |  7844 | TURNER | 1500.00 |    0.00 |
    +-------+--------+---------+---------+
```

and 并且

查询工作岗位是MANAGER并且工资大于2500的员工信息？

```
    select
        empno,ename,job,sal
    from
        emp
    where
        job = 'MANAGER' and sal > 2500;

    +-------+-------+---------+---------+
    | empno | ename | job     | sal     |
    +-------+-------+---------+---------+
    |  7566 | JONES | MANAGER | 2975.00 |
    |  7698 | BLAKE | MANAGER | 2850.00 |
    +-------+-------+---------+---------+
```

or 或者

查询工作岗位是MANAGER和SALESMAN的员工？

```
    select empno,ename,job from emp where job = 'MANAGER';
    select empno,ename,job from emp where job = 'SALESMAN';

    select
        empno,ename,job
    from
        emp
    where
        job = 'MANAGER' or job = 'SALESMAN';

    +-------+--------+----------+
    | empno | ename  | job      |
    +-------+--------+----------+
    |  7499 | ALLEN  | SALESMAN |
    |  7521 | WARD   | SALESMAN |
    |  7566 | JONES  | MANAGER  |
    |  7654 | MARTIN | SALESMAN |
```

```
|  7698 | BLAKE   | MANAGER   |
|  7782 | CLARK   | MANAGER   |
|  7844 | TURNER  | SALESMAN  |
+-------+---------+-----------+
```

and和or同时出现的话，有优先级问题吗？

查询工资大于2500，并且部门编号为10或20部门的员工？

```
select
    *
from
    emp
where
    sal > 2500 and deptno = 10 or deptno = 20;
```

分析以上语句的问题？

and优先级比or高。

以上语句会先执行and，然后执行or。

以上这个语句表示什么含义？

找出工资大于2500并且部门编号为10的员工，或者20部门所有员工找出来。

```
select
    *
from
    emp
where
    sal > 2500 and (deptno = 10 or deptno = 20);
```

and和or同时出现，and优先级较高。如果想让or先执行，需要加“小括号”

★以后在开发中，如果不确定优先级，就加小括号就行了。

in 包含，相当于多个 or （not in 不在这个范围中） ！！不是区间

查询工作岗位是MANAGER和SALESMAN的员工？

```
select empno,ename,job from emp where job = 'MANAGER' or job = 'SALESMAN';
select empno,ename,job from emp where job in('MANAGER', 'SALESMAN');
```

```
+-------+--------+----------+
| empno | ename  | job      |
+-------+--------+----------+
|  7499 | ALLEN  | SALESMAN |
|  7521 | WARD   | SALESMAN |
|  7566 | JONES  | MANAGER  |
|  7654 | MARTIN | SALESMAN |
|  7698 | BLAKE  | MANAGER  |
|  7782 | CLARK  | MANAGER  |
|  7844 | TURNER | SALESMAN |
+-------+--------+----------+
```

注意：in不是一个区间。in后面跟的是具体的值。

查询薪资是800和5000的员工信息？

```
select ename,sal from emp where sal = 800 or sal = 5000;
select ename,sal from emp where sal in(800, 5000); //这个不是表示800到5000都找出来。
```

```
+-------+---------+
| ename | sal     |
+-------+---------+
| SMITH |  800.00 |
| KING  | 5000.00 |
+-------+---------+
```

```
select ename,sal from emp where sal in(800, 5000, 3000);

// not in 表示不在这几个值当中的数据。
select ename,sal from emp where sal not in(800, 5000, 3000);
```

```
+--------+---------+
| ename  | sal     |
```

```
+--------+---------+
| ALLEN  | 1600.00 |
| WARD   | 1250.00 |
| JONES  | 2975.00 |
| MARTIN | 1250.00 |
| BLAKE  | 2850.00 |
| CLARK  | 2450.00 |
| TURNER | 1500.00 |
| ADAMS  | 1100.00 |
| JAMES  |  950.00 |
| MILLER | 1300.00 |
+--------+---------+
```

not 可以取非，主要用在 is 或 in 中

```
is null
is not null
in
not in
```

like

称为模糊查询，支持%或下划线匹配

%匹配任意多个字符

下划线：任意一个字符。

（%是一个特殊的符号，_ 也是一个特殊符号）

找出名字中含有O的？

```
mysql> select ename from emp where ename like '%O%';
+-------+
| ename |
+-------+
| JONES |
| SCOTT |
| FORD  |
```

```
    +-------+

    找出名字以T结尾的？
        select ename from emp where ename like '%T';
            找出名字以K开始的？
        select ename from emp where ename like 'K%';

    找出第二个字每是A的？
        select ename from emp where ename like
'_A%';

    找出第三个字母是R的？
        select ename from emp where ename like
'__R%';

    t_student学生表
    name字段

----------------------

    zhangsan
    lisi
    wangwu
    zhaoliu
    jack_son

    找出名字中有“_”的？
        select name from t_student where name like
'%_%'; //这样不行。

        mysql> select name from t_student where name
like '%\_%'; // \转义字符。
        +----------+
        | name     |
        +----------+
        | jack_son |
        +----------+
```

2.8 排序 order by

```
select
   ename,sal
from
   emp
order by
   sal; // 默认是升序！！

//指定降序
select
    ename,sal
from
    emp
order by
    sal desc;
//指定升序
select
    ename,sal
from
    emp
order by
    sal asc;

//多个字段排序
select
    ename,sal
from
    emp
order by
    sal asc, ename asc; // sal在前，起主导，只有sal相等的时候，才会考虑启用ename排序

//根据字段的位置也可以排序
select ename,sal from emp order by 2; // 2表示第二列。第二列是sal
```

3、数据处理函数

单行处理函数的特点：一个输入对应一个输出。

和单行处理函数相对的是：多行处理函数。（多行处理函数特点：多个输入，对应1个输出！）

常见的单行处理函数：

lower 转换小写

```
mysql> select lower(ename) as ename from emp;
```

upper 转换大写

```
mysql> select upper(name) as name from t_student;
```

substr(被截取的字符串, 起始下标,截取的长度) 取子串

```
select substr(ename, 1, 1) as ename from emp;
```

concat(字符串1， 字符串2) 连接两个字符串

```
首字母大写
select
concat(upper(substr(name,1,1)),substr(name,2,length(
name) - 1)) as result from t_student;

select concat(empno,ename) from emp;
```

length 取长度

```
select length(ename) enamelength from emp;
```

trim 去空格

```
select * from emp where ename = trim('   KING');
```

str_to_date 将字符串转换成日期
date_format 格式化日期
format 设置千分位

case..when..then..when..then..else..end 当case指向的字段 不同的情况是应该做什么操作

```
当员工的工作岗位是MANAGER的时候，工资上调10%，当工作岗位是SALESMAN的时候，工资上调50%,其它正常。
（注意：不修改数据库，只是将查询结果显示为工资上调）
select
    ename,
    job,
    sal as oldsal,
    (case job when 'MANAGER' then sal*1.1 when 'SALESMAN' then sal*1.5 else sal end) as newsal
from
    emp;
```

round(数，保留的小数位数) 四舍五入。如果保留的小数位数小于0，就向前延

```
select round(1236.567, 0) as result from emp; //保留整数位。
```

rand() 生成0-1之间的随机数（小数，如果要取整则要加round函数)

```
select round(rand()*100,0) from emp; // 100以内的随机数
```

ifnull 可以将 null 转换成一个具体值

```
select ename, (sal + ifnull(comm, 0)) * 12 as yearsal from emp;//如果comm为null则补为0
```

4、分组函数

多行处理函数的特点：输入多行，最终输出一行。

共五个函数：

```
count 计数
sum 求和
avg 平均值
max 最大值
min 最小值
```

注意：

分组函数在使用的时候必须先进行分组，然后才能用。

如果你没有对数据进行分组，整张表默认为一组。

```
select max(sal) from emp;
select min(sal) from emp;
select sum(sal) from emp;
select avg(sal) from emp;
select count(ename) from emp;
```

注意：

分组函数自动忽略null，不需要提前对NULL进行处理

count(*)统计表当中的总行数。（只要有一行数据count则++）

count(具体字段)表示统计该字段下所有不为NULL的元素的总数

分组函数不能够直接使用在where子句中

所有的分组函数可以组合起来一起用

5、分组查询

在实际的应用中，可能有这样的需求，需要先进行分组，然后对每一组的数据进行操作。这个时候我们需要使用分组查询。

```
select
    ...
from
    ...
group by
    ...
```

执行顺序

1.from
2. where
3. group by
4. select
5. order by

5.1为什么分组函数不能直接使用在where后面

```
select ename,sal from emp where sal > min(sal);//报错。
```

因为**分组函数在使用的时候必须先分组之后才能使用。**
where执行的时候，还没有分组。所以where后面不能出现分组函数。

```
select sum(sal) from emp;
```

这个没有分组，为啥sum()函数可以用呢？因为select在group by之后执行。

按照工作岗位分组，然后对工资求和。

```
select
    job,sum(sal)
```

```
from
    emp
group by
    job;

+-----------+----------+
| job       | sum(sal) |
+-----------+----------+
| ANALYST   |  6000.00 |
| CLERK     |  4150.00 |
| MANAGER   |  8275.00 |
| PRESIDENT |  5000.00 |
| SALESMAN  |  5600.00 |
+-----------+----------+
```

以上这个语句的执行顺序?

先从emp表中查询数据。 from

根据job字段进行分组。group by

然后对每一组的数据进行sum(sal)

重点结论:

在一条select语句当中，如果有group by语句的话，select后面只能跟：参加分组的字段，以及分组函数。其它的一律不能跟。

5.2 可以对多个字段联合分组查询

找出“每个部门，不同工作岗位”的最高薪资

两个字段联合成1个字段看。（两个字段联合分组）

```
    select
        deptno, job, max(sal)
    from
        emp
```

```
    group by
        deptno, job;

    +--------+-----------+----------+
    | deptno | job       | max(sal) |
    +--------+-----------+----------+
    |     10 | CLERK     |  1300.00 |
    |     10 | MANAGER   |  2450.00 |
    |     10 | PRESIDENT |  5000.00 |
    |     20 | ANALYST   |  3000.00 |
    |     20 | CLERK     |  1100.00 |
    |     20 | MANAGER   |  2975.00 |
    |     30 | CLERK     |   950.00 |
    |     30 | MANAGER   |  2850.00 |
    |     30 | SALESMAN  |  1600.00 |
    +--------+-----------+----------+
```

5.3 使用having可以对分完组之后的数据进一步过滤

having不能单独使用，having不能代替where，having必须和group by联合使用。

找出每个部门最高薪资，要求显示最高薪资大于3000的

```
select
deptno,max(sal)
from
emp
group by
deptno
having
max(sal) > 3000;

+--------+----------+
| deptno | max(sal) |
```

```
+--------+----------+
|     10 |  5000.00 |
+--------+----------+
```

5.4 没办法使用where再用having

where和having，优先选择where，where实在完成不了了，再选择having。

如：找出每个部门平均薪资，要求显示平均薪资高于2500的。

```
select
deptno,avg(sal)
from
emp
group by
deptno
having
avg(sal) > 2500;

+--------+-------------+
| deptno | avg(sal)    |
+--------+-------------+
|     10 | 3208.333333 |
+--------+-------------+
```

5.5 查询结果去重

表数据不会被修改，只是查询结果去重。

去重需要使用一个关键字：distinct

```
mysql> select distinct job from emp;
+-----------+
| job       |
+-----------+
```

```
| CLERK     |
| SALESMAN  |
| MANAGER   |
| ANALYST   |
| PRESIDENT |
+-----------+

// 以下编写是错误的，语法错误。
// distinct只能出现在所有字段的最前方。
mysql> select ename,distinct job from emp;

// distinct出现在job,deptno两个字段之前，表示两个字段联合起来去重。
mysql> select distinct job,deptno from emp;
+-----------+--------+
| job       | deptno |
+-----------+--------+
| CLERK     |     20 |
| SALESMAN  |     30 |
| MANAGER   |     20 |
| MANAGER   |     30 |
| MANAGER   |     10 |
| ANALYST   |     20 |
| PRESIDENT |     10 |
| CLERK     |     30 |
| CLERK     |     10 |
+-----------+--------+

统计一下工作岗位的数量？
    select count(distinct job) from emp;
    +--------------------+
    | count(distinct job) |
    +--------------------+
    |                  5 |
    +--------------------+
```

6、连接查询（多表联合查询）

6.1连接查询的分类

根据语法的年代分类：

SQL92：1992年的时候出现的语法

SQL99：1999年的时候出现的语法

我们这里重点学习SQL99.(这个过程中简单演示一个SQL92的例子)

根据表连接的方式分类：

内连接：

等值连接

非等值连接

自连接

外连接：

左外连接（左连接）

右外连接（右连接）

全连接（不讲）

6.2 笛卡尔积现象

两张表进行连接查询时，没有任何条件的限制

两张表连接没有任何条件限制：

```
select ename,dname from emp, dept;
+--------+------------+
| ename  | dname      |
+--------+------------+
| SMITH  | ACCOUNTING |
| SMITH  | RESEARCH   |
| SMITH  | SALES      |
| SMITH  | OPERATIONS |
| ALLEN  | ACCOUNTING |
```

```
| ALLEN  | RESEARCH   |
| ALLEN  | SALES      |
| ALLEN  | OPERATIONS |
...
56 rows in set (0.00 sec)
//emp表数据条数：14 dept表条数：4   14 * 4 = 56
```

当两张表进行连接查询，没有任何条件限制的时候，最终查询结果条数，是
两张表条数的乘积，这种现象被称为：笛卡尔积现象。（笛卡尔发现的，这是
一个数学现象。）

避免笛卡尔积现象

连接时加条件，满足这个条件的记录被筛选出来！

```sql
select
ename,dname
from
emp, dept
where
emp.deptno = dept.deptno;
```

```sql
select
    emp.ename,dept.dname
from
    emp, dept
where
    emp.deptno = dept.deptno;

// 表起别名。很重要。效率问题。
select
    e.ename,d.dname
from
    emp e, dept d
where
```

```
    e.deptno = d.deptno; //SQL92语法。

+--------+------------+
| ename  | dname      |
+--------+------------+
| CLARK  | ACCOUNTING |
| KING   | ACCOUNTING |
| MILLER | ACCOUNTING |
| SMITH  | RESEARCH   |
| JONES  | RESEARCH   |
| SCOTT  | RESEARCH   |
| ADAMS  | RESEARCH   |
| FORD   | RESEARCH   |
| ALLEN  | SALES      |
| WARD   | SALES      |
| MARTIN | SALES      |
| BLAKE  | SALES      |
| TURNER | SALES      |
| JAMES  | SALES      |
+--------+------------+
```

思考：最终查询的结果条数是14条，但是匹配的过程中，匹配的次数减少了吗？

还是56次，只不过进行了四选一。次数没有减少。

注意：通过笛卡尔积现象得出，表的连接次数越多效率越低，尽量避免表的
连接次数。

6.3 内连接-等值连接 inner join...on...

两张表通过不同字段的等于关系连接

案例：查询每个员工所在部门名称，显示员工名和部门名
emp e和dept d表进行连接。条件是：e.deptno = d.deptno

SQL92语法：

```
select
e.ename,d.dname
from
emp e, dept d
where
e.deptno = d.deptno;
```

sql92的缺点：结构不清晰，表的连接条件，和后期进一步筛选的条件，都放到了where后面。

SQL99语法：

```
select
e.ename,d.dname
from
emp e
join
dept d
on
e.deptno = d.deptno;
```

```
//inner可以省略（带着inner可读性更好！！！一眼就能看出来是内连接）
select
    e.ename,d.dname
from
    emp e
inner join
    dept d
on
    e.deptno = d.deptno; // 条件是等量关系，所以被称为等值连接。
```

sql99优点：表连接的条件是独立的，连接之后，如果还需要进一步筛选，再往后继续添加where

SQL99语法：

```
select
    ...
from
    a
join
    b
on
    a和b的连接条件
where
    筛选条件
```

6.4 内连接-非等值连接

两张表不通过不同字段的等于关系连接(条件不是一个等量关系，称为非等值连接。)

案例：找出每个员工的薪资等级，要求显示员工名、薪资、薪资等级

```
select
    e.ename, e.sal, s.grade
from
    emp e
inner join
    salgrade s
on
    e.sal between s.losal and s.hisal;
```

```
+--------+---------+-------+
| ename  | sal     | grade |
+--------+---------+-------+
| SMITH  |  800.00 |     1 |
| ALLEN  | 1600.00 |     3 |
| WARD   | 1250.00 |     2 |
| JONES  | 2975.00 |     4 |
```

```
| MARTIN | 1250.00 |      2 |
| BLAKE  | 2850.00 |      4 |
| CLARK  | 2450.00 |      4 |
| SCOTT  | 3000.00 |      4 |
| KING   | 5000.00 |      5 |
| TURNER | 1500.00 |      3 |
| ADAMS  | 1100.00 |      1 |
| JAMES  |  950.00 |      1 |
| FORD   | 3000.00 |      4 |
| MILLER | 1300.00 |      2 |
+--------+---------+--------+
```

6.5 内连接-自连接

案例：查询员工的上级领导，要求显示员工名和对应的领导名？

```
//技巧：一张表看成两张表
select
    a.ename as '员工名', b.ename as '领导名'
from
    emp a
join
    emp b
on
    a.mgr = b.empno; //员工的领导编号 = 领导的员工编号

+--------+--------+
| 员工名 | 领导名|
+--------+--------+
| SMITH  | FORD   |
| ALLEN  | BLAKE  |
| WARD   | BLAKE  |
| JONES  | KING   |
| MARTIN | BLAKE  |
| BLAKE  | KING   |
| CLARK  | KING   |
| SCOTT  | JONES  |
```

```
| TURNER | BLAKE   |
| ADAMS  | SCOTT   |
| JAMES  | BLAKE   |
| FORD   | JONES   |
| MILLER | CLARK   |
+--------+---------+
```

13条记录，没有KING。《内连接》

以上就是内连接中的：自连接，技巧：一张表看做两张表。

6.6 外连接

内连接：（A和B连接，AB两张表没有主次关系。平等的。）

```
外连接（右外连接）：
select
    e.ename,d.dname
from
    emp e
right join
    dept d
on
    e.deptno = d.deptno;

// outer是可以省略的，带着可读性强。
select
    e.ename,d.dname
from
    emp e
right outer join
    dept d
on
    e.deptno = d.deptno;
```

right代表什么：表示将join关键字右边的这张表看成主表，主要是为了将
这张表的数据全部查询出来，捎带着关联查询左边的表。
在外连接当中，两张表连接，产生了主次关系。如果主表中的数据副表中没有，也会查出来并且副表值设为NULL

外连接的查询结果条数一定是 >= 内连接的查询结果条数

```
外连接（左外连接）：
select
    e.ename,d.dname
from
    dept d
left join
    emp e
on
    e.deptno = d.deptno;

// outer是可以省略的，带着可读性强。
select
    e.ename,d.dname
from
    dept d
left outer join
    emp e
on
    e.deptno = d.deptno;
```

带有right的是右外连接，又叫做右连接。
带有left的是左外连接，又叫做左连接。
任何一个右连接都有左连接的写法。
任何一个左连接都有右连接的写法。

6.7 两张以上的表的连接

语法:

```
select
...
from
a
join
b
on
a和b的连接条件
join
c
on
a和c的连接条件
right join
d
on
a和d的连接条件
```

一条SQL语句中内连接和外连接可以混合。都可以出现!

案例：找出每个员工的部门名称以及工资等级，还有上级领导，
要求显示员工名、领导名、部门名、薪资、薪资等级？

```
select
    e.ename,e.sal,d.dname,s.grade,l.ename
from
    emp e
join
    dept d
on
    e.deptno = d.deptno
join
    salgrade s
on
```

```
    e.sal between s.losal and s.hisal
left join
    emp l
on
    e.mgr = l.empno;
```

```
+--------+---------+------------+-------+-------+
| ename  | sal     | dname      | grade | ename |
+--------+---------+------------+-------+-------+
| SMITH  |  800.00 | RESEARCH   |     1 | FORD  |
| ALLEN  | 1600.00 | SALES      |     3 | BLAKE |
| WARD   | 1250.00 | SALES      |     2 | BLAKE |
| JONES  | 2975.00 | RESEARCH   |     4 | KING  |
| MARTIN | 1250.00 | SALES      |     2 | BLAKE |
| BLAKE  | 2850.00 | SALES      |     4 | KING  |
| CLARK  | 2450.00 | ACCOUNTING |     4 | KING  |
| SCOTT  | 3000.00 | RESEARCH   |     4 | JONES |
| KING   | 5000.00 | ACCOUNTING |     5 | NULL  |
| TURNER | 1500.00 | SALES      |     3 | BLAKE |
| ADAMS  | 1100.00 | RESEARCH   |     1 | SCOTT |
| JAMES  |  950.00 | SALES      |     1 | BLAKE |
| FORD   | 3000.00 | RESEARCH   |     4 | JONES |
| MILLER | 1300.00 | ACCOUNTING |     2 | CLARK |
+--------+---------+------------+-------+-------+
```

7、子查询

7.1 什么是子查询

select语句中嵌套select语句，被嵌套的select语句称为子查询。

```
select
    ..(select).
from
    ..(select).
where
    ..(select).
```

7.2 where子句中的子查询

案例：找出比最低工资高的员工姓名和工资?

```
select ename,sal from emp where sal > (select
min(sal) from emp);
```

7.3 from子句中的子查询

注意：from后面的子查询，可以将子查询的查询结果当做一张临时表。（技巧）

案例：找出每个岗位的平均工资的薪资等级。

```
select
    t.*, s.grade
from
    (select job,avg(sal) as avgsal from emp group by
job) t
join
    salgrade s
on
    t.avgsal between s.losal and s.hisal;

+-----------+-------------+-------+
| job       | avgsal      | grade |
+-----------+-------------+-------+
| CLERK     | 1037.500000 |     1 |
| SALESMAN  | 1400.000000 |     2 |
```

```
| ANALYST   | 3000.000000 |     4 |
| MANAGER   | 2758.333333 |     4 |
| PRESIDENT | 5000.000000 |     5 |
+-----------+-------------+-------+
```

7.4 select后面出现的子查询

案例：找出每个员工的部门名称，要求显示员工名，部门名？

```
select
e.ename,e.deptno,(select d.dname from dept d where
e.deptno = d.deptno) as dname
from
emp e;
```

```
+--------+--------+------------+
| ename  | deptno | dname      |
+--------+--------+------------+
| SMITH  |     20 | RESEARCH   |
| ALLEN  |     30 | SALES      |
| WARD   |     30 | SALES      |
| JONES  |     20 | RESEARCH   |
| MARTIN |     30 | SALES      |
| BLAKE  |     30 | SALES      |
| CLARK  |     10 | ACCOUNTING |
| SCOTT  |     20 | RESEARCH   |
| KING   |     10 | ACCOUNTING |
| TURNER |     30 | SALES      |
| ADAMS  |     20 | RESEARCH   |
| JAMES  |     30 | SALES      |
| FORD   |     20 | RESEARCH   |
| MILLER |     10 | ACCOUNTING |
+--------+--------+------------+

//以下是错误的：ERROR 1242 (21000): Subquery returns
more than 1 row
select
```

```
    e.ename,e.deptno,(select dname from dept) as
dname
from
    emp e;
```

注意：对于select后面的子查询来说，这个子查询只能一次返回1条结果，
多于1条，就报错了。！

8、合并查询union

案例：查询工作岗位是MANAGER和SALESMAN的员工?

```
select ename,job from emp where job = 'MANAGER' or
job = 'SALESMAN';
select ename,job from emp where job
in('MANAGER','SALESMAN');
+--------+----------+
| ename  | job      |
+--------+----------+
| ALLEN  | SALESMAN |
| WARD   | SALESMAN |
| JONES  | MANAGER  |
| MARTIN | SALESMAN |
| BLAKE  | MANAGER  |
| CLARK  | MANAGER  |
| TURNER | SALESMAN |
+--------+----------+
```

```
select ename,job from emp where job = 'MANAGER'
union
select ename,job from emp where job = 'SALESMAN';
+--------+----------+
| ename  | job      |
+--------+----------+
| JONES  | MANAGER  |
| BLAKE  | MANAGER  |
```

```
| CLARK  | MANAGER  |
| ALLEN  | SALESMAN |
| WARD   | SALESMAN |
| MARTIN | SALESMAN |
| TURNER | SALESMAN |
+--------+----------+
```

union的效率要高一些。对于表连接来说，每连接一次新表，则匹配的次数满足笛卡尔积，成倍的翻。。。
但是union可以减少匹配的次数。在减少匹配次数的情况下，还可以完成两个结果集的拼接。